Extracting Text Between Quotes: A Google Sheets Tutorial Using Regular Expressions

Authored by

Mohammed looti

November 12, 2025

Harnessing Regular Expressions for Precise Text Extraction in Google Sheets

In modern data analysis and cleaning workflows, the ability to isolate specific pieces of information from complex text strings is paramount. When working within Google Sheets, analysts frequently encounter raw data--often imported from database logs, system outputs, or user entries--where critical values are deliberately enclosed within quotation marks. These quoted segments typically represent essential data points such as unique IDs, descriptive names, or coded variables. To efficiently tackle this common challenge and eliminate manual data scrubbing, Google Sheets offers access to the advanced functionality of Regular Expression syntax through its powerful built-in functions.

This comprehensive guide is designed to demystify the process, detailing the exact formulas required to reliably capture and isolate text that is enclosed within either double quotes (`" "`) or single quotes (`' '`). Our methodology centers exclusively on the use of the **REGEXEXTRACT** function, a highly versatile tool tailored for pattern-based data retrieval. Mastering the construction of these specific formulas is crucial for anyone looking to automate their data preparation tasks, guaranteeing that extraneous surrounding text is immediately discarded and only the necessary, clean quoted content remains for subsequent reporting or analysis. We will meticulously explore the underlying mechanics of the **Regular Expression** patterns used, providing a clear explanation of how they achieve such targeted and error-free extraction.

Implementing the Core REGEXEXTRACT Formulas

The core principle behind successful quoted text extraction relies entirely on the structure of the **REGEXEXTRACT** function. This function mandates two essential arguments: first, the cell reference containing the source data string, and second, the specific regular expression pattern that defines the text intended for capture. Crucially, this pattern must incorporate a **capturing group**--defined using parentheses--which explicitly instructs the function exactly what subset of the successful match should be returned as the final output, thereby excluding the quotation marks themselves.

We present two primary formulas below, each carefully tailored to handle the distinct requirements of double versus single quotation marks. For all demonstrations, we will assume that the source data requiring extraction is consistently located in cell **A2**. These formulas are intentionally designed to be robust and effective, regardless of the length or complexity of the text surrounding the delimiters.

Method 1: Extracting Content Delimited by Double Quotes

=REGEXEXTRACT(A2,"""(.*)""")

The pattern `"""(.*)"""` is specifically engineered to handle the challenges of double-quote extraction within the constraints of Google Sheets formula parsing. The necessity of using **triple double quotes** at both the beginning and end of the pattern string is a key concept here. The two inner double quotes are the actual regex delimiters we are searching for, but they must be properly escaped within the outer double quotes that define the string argument for the formula itself. The centerpiece of the expression, `(.*)`, functions as the capturing group, designed to isolate and return any sequence of characters found between those defined delimiters.

Method 2: Extracting Content Delimited by Single Quotes

=REGEXEXTRACT(A2,"'(.*)'")

In scenarios where source data utilizes single quotes (apostrophes) as delimiters, the required pattern structure is notably simpler. The expression `"'(.*)'"` requires only four quotes in total: the outer double quotes define the string argument in the spreadsheet formula, and the inner single quotes define the literal delimiters for the Regular Expression engine. This reduction in complexity stems from the fact that single quotes do not necessitate the same stringent level of escaping within the standard Google Sheets formula string environment as double quotes do. Both methods showcase the efficiency and adaptability of the REGEXEXTRACT function across diverse data formatting styles.

Deconstructing the Power of the Regular Expression Capturing Group

The extraordinary effectiveness of these extraction formulas is rooted directly in the efficient design of the Regular Expression pattern itself, particularly the capturing group denoted by `(.*)`. A thorough understanding of this concise yet powerful sequence is fundamental for anyone aspiring to master advanced text manipulation within spreadsheet environments. The primary, indispensable role of the parentheses is to explicitly guide the REGEXEXTRACT function, signaling precisely which matched segment of the overall pattern should be returned to the cell as the result.

The components enclosed within the parentheses are standard, widely-used elements found in virtually all modern Regular Expression implementations, adhering to the RE2 syntax used by Google Sheets:

The `.` (dot) is recognized as the fundamental **wildcard character**. In this context, it is designed to match any single character available, typically excluding newline characters in standard RE2 behavior.

The `*` (asterisk) serves as a powerful **quantifier**. It dictates that the preceding element--which is the wildcard dot--must be matched zero or more times. This mechanism imparts a "greedy" nature

to the match, ensuring that the entire span of text found between the opening and closing quotation marks is captured, irrespective of the length of the string.

When combined, the sequence `(.*)` essentially translates to the instruction: "Capture everything." By strategically positioning this capturing group between the necessary quotation mark delimiters (be they `"` or `'`), the formula successfully isolates the content of specific interest. Because **REGEXEXTRACT** is inherently programmed to return only the content of the first capturing group encountered, the quotation marks themselves are utilized strictly for defining the boundary conditions and are automatically excluded from the final cell output, guaranteeing perfectly clean, extracted data without the delimiters.

Practical Application: Extracting Data Delimited by Double Quotes

To clearly demonstrate the implementation of Method 1, let us examine a typical scenario involving athlete records. Imagine a dataset where critical identifying information, such as unique nicknames or specific team designations, is consistently wrapped in double quotes. The essential step for data normalization or focused analysis is the isolation of this specific quoted data into its own dedicated column. Assume Column A contains the raw, combined data, as visualized in the accompanying screenshot below:

	A	B	C
1	**Name**		
2	Andy "Super" James		
3	Bob "Runaway"Smith		
4	Chad "Speedy" Anderson		
5	Doug "King" Miller		
6	Ethan "Tall Guy" Wilks		
7	Frank "The Ghost" Tomms		
8	Greg "Thriller" Dodson		
9			
10			
11			
12			
13			

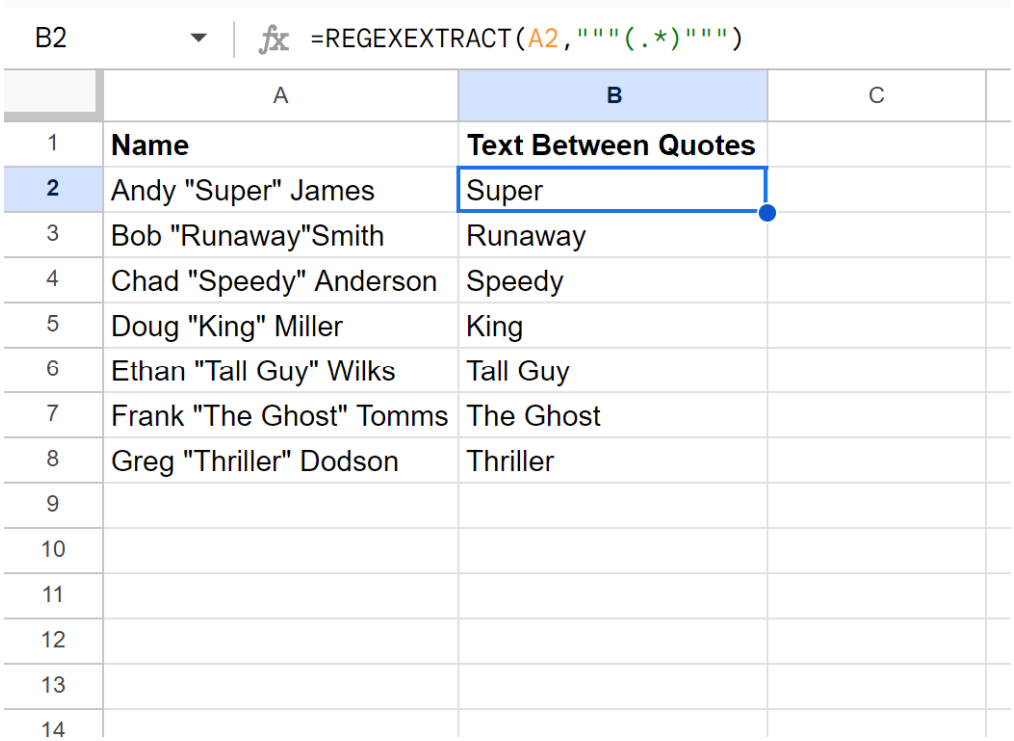
Our primary objective is to swiftly move the content found exclusively within the double quotes from Column A to Column B. This process must be highly scalable, effectively eliminating the cumbersome requirement for manual inspection and cleaning across potentially thousands of rows.

By leveraging the REGEXEXTRACT function, we guarantee consistent application of the extraction logic across the entire data range with maximum speed and reliability.

We initiate the process by inputting the formula specifically designed for double-quote extraction into cell **B2**, ensuring it references the corresponding source text located in cell **A2**:

=REGEXEXTRACT(A2,"""(.*)""")

Upon execution, cell B2 will immediately display the perfectly isolated, clean text. To efficiently complete the task for the remainder of the dataset, we simply utilize the powerful **fill handle** functionality inherent in Google Sheets. Clicking and dragging the formula downward through all populated cells in Column B automatically adjusts the relative cell references (e.g., A2 intelligently shifts to A3, A4, and so on), applying the precise extraction logic seamlessly to every single row in the list.



	A	B	C
1	**Name**	**Text Between Quotes**	
2	Andy "Super" James	Super	
3	Bob "Runaway"Smith	Runaway	
4	Chad "Speedy" Anderson	Speedy	
5	Doug "King" Miller	King	
6	Ethan "Tall Guy" Wilks	Tall Guy	
7	Frank "The Ghost" Tomms	The Ghost	
8	Greg "Thriller" Dodson	Thriller	
9			
10			
11			
12			
13			
14			

The final outcome, vividly illustrated in the image above, confirms that Column B now holds the clean, accurately extracted data, completely devoid of the surrounding noise and, most importantly, the double quotation marks themselves. This successful result validates the formula's high effectiveness in segmenting complex data based on double-quote delimiters, providing an indispensable tool for data preparation.

Targeted Extraction Using Single Quotes in Technical Data

Often, data imported from highly technical environments--such as output from SQL queries, specific Python dictionaries, or system configuration files--will utilize single quotes (apostrophes) as the primary text delimiter, presenting a slight variation from standard text documents. Successfully handling this format requires a minor adjustment to the **Regular Expression** pattern while ensuring that the core efficiency of the extraction method is fully maintained. Consider the following example dataset where the target textual data is consistently enclosed within single quotes:

	A	B	C
1	**Name**		
2	Andy 'Super' James		
3	Bob 'Runaway'Smith		
4	Chad 'Speedy' Anderson		
5	Doug 'King' Miller		
6	Ethan 'Tall Guy' Wilks		
7	Frank 'The Ghost' Tomms		
8	Greg 'Thriller' Dodson		
9			
10			
11			
12			
13			
14			
15			

Our objective remains the same: the precise isolation and extraction of the text residing strictly between the single quotes for every corresponding entry in Column A. This task demands accuracy but benefits significantly from the simplified escaping rules associated with single quotes within the Google Sheets environment, as discussed earlier.

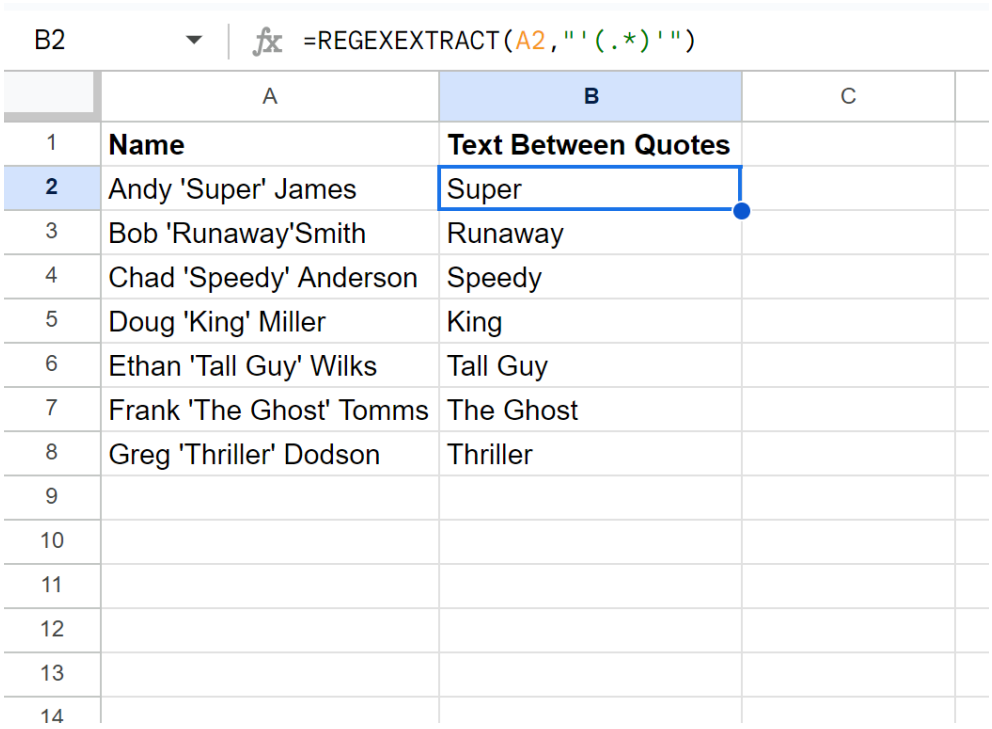
We initiate this specific process by entering the single-quote extraction formula into cell **B2**, ensuring it correctly references cell **A2**:

=REGEXEXTRACT(A2,"'(.*)'")

This streamlined formula effectively utilizes the pattern `"'(.*)'"`. It efficiently locates the opening

single quote, captures all intermediate content via the powerful `(.*)` capturing group, and terminates the match precisely at the closing single quote. Once the formula is correctly entered and confirmed, the desired extracted text immediately populates cell B2.

Consistent with the previous example, the extraction is efficiently completed for the entire column by employing the fill handle. This action propagates the formula down the column, ensuring that every single row is processed with absolute accuracy using the predefined **Regular Expression** pattern, regardless of data complexity.



	A	B	C
1	**Name**	**Text Between Quotes**	
2	Andy 'Super' James	Super	
3	Bob 'Runaway'Smith	Runaway	
4	Chad 'Speedy' Anderson	Speedy	
5	Doug 'King' Miller	King	
6	Ethan 'Tall Guy' Wilks	Tall Guy	
7	Frank 'The Ghost' Tomms	The Ghost	
8	Greg 'Thriller' Dodson	Thriller	
9			
10			
11			
12			
13			
14			

The final output clearly visible in Column B, demonstrated above, confirms the perfect extraction of text delimited exclusively by single quotes. Regardless of the specific type of quotation marks employed in your raw source data, leveraging the correct **REGEXEXTRACT** formula ensures accurate, reliable, and highly scalable data cleaning, thereby significantly enhancing your spreadsheet data preparation capabilities.

Advanced REGEX Techniques and Related String Functions

While **REGEXEXTRACT** is the specialized function for isolating specific content, its overall utility within the data cleaning ecosystem is often substantially amplified when integrated or considered alongside other powerful string manipulation functions available in Google Sheets. For handling more intricate data cleaning requirements--such as dealing with source cells that contain multiple instances of quotes, or navigating complex nested quote structures--simple `(.*)` patterns may

prove insufficient. Such complex scenarios typically demand advanced **Regular Expression** techniques. For example, if the requirement is to extract only the very first quoted segment from a long string containing several distinct quoted sections, implementing non-greedy matching using the quantifier `.*?` becomes necessary to prevent the match from spanning across multiple delimiters.

Furthermore, mastering the entire suite of regex-based functions provides a comprehensive toolkit for managing virtually any format of textual data encountered in spreadsheet operations. For rapid data validation, the `REGEXMATCH` function can be deployed to swiftly identify which cells conform to (or deviate from) a pattern, such as containing quoted text. Conversely, `REGEXREPLACE` offers the capability to remove the quotes entirely from the original source string, serving a different, yet equally vital, data cleaning purpose than mere extraction. Developing proficiency across these related functions ensures you have the capability to handle textual data transformation robustly and efficiently.

To further build upon the foundational knowledge of **Regular Expression** usage demonstrated in this guide, consider exploring the following advanced tutorials:

Exploring the capabilities of **REGEXMATCH** for dynamic pattern validation and quality checks.

Utilizing **REGEXREPLACE** for targeted text substitution, removal, and complex cleaning operations.

Combining Array formulas with text functions for highly dynamic, column-wide processing of data changes.